



HiEulerOS

开发调试 FAQ

文档版本 01

发布日期 2024-04-26

版权所有 © 海思技术有限公司2024。保留一切权利。

非经本公司书面许可，任何单位和个人不得擅自摘抄、复制本文档内容的部分或全部，并不得以任何形式传播。

商标声明



HISILICON、海思和其他海思商标均为海思技术有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受海思公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，海思公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

海思技术有限公司

地址：上海市青浦区虹桥港路2号101室 邮编：201721

网址：<https://www.hisilicon.com/cn/>

客户服务邮箱：support@hisilicon.com



前言

概述

本文档为使用海思Hi3093芯片，基于SDK包进行二次开发的程序员而写，目的是解答您开发过程中可能遇到的问题。

产品版本

与本文档相对应的产品版本如下。

产品名称	产品版本
Hi3093	-

读者对象

本文档（本指南）主要适用于以下工程师：

- 技术支持工程师
- 软件开发工程师

符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
 危险	表示如不避免则将会导致死亡或严重伤害的具有高等级风险的危害。
 警告	表示如不避免则可能导致死亡或严重伤害的具有中等级风险的危害。
 注意	表示如不避免则可能导致轻微或中度伤害的具有低等级风险的危害。



符号	说明
<div>须知</div>	用于传递设备或环境安全警示信息。如不可避免则可能会导致设备损坏、数据丢失、设备性能降低或其它不可预知的结果。 “须知”不涉及人身伤害。
<div>说明</div>	对正文中重点信息的补充说明。 “说明”不是安全警示信息，不涉及人身、设备及环境伤害信息。

修订记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有文档版本的更新内容。

日期	版本	修订说明
2023-10-31	00B01	第1次临时版本发布。
2024-04-26	01	第1次正式版本发布。



目录

前言.....	i
1 添加驱动指导.....	1
2 可用外设资源配置.....	2
3 I2C 、SPI、DMAC 使用 FAQ.....	4
4 NET 使用 FAQ.....	5
5 IPC 性能测试方法.....	6
5.1 Baremetal 测试方法.....	7
5.2 Libck 测试方法.....	8
5.3 UniProton 测试方法.....	9
6 虚拟串口使用.....	12
7 下电释放资源.....	13
8 串口打印.....	14
9 openEuler_tcpdump 编译.....	15
10 混合部署实时侧驱动测试方法.....	16
10.1 混合部署 Baremetal 侧驱动测试.....	16
10.2 混合部署 Libck 侧驱动测试.....	16
11 实时侧驱动数据传输接口地址保护开关.....	18
12 uboot 控制台读写 eMMC 和 SFC.....	19
12.1 sfc 读写.....	19
12.2 eMMC 读写.....	19
13 其他测试.....	20



插图目录

图 2-1 dts 转换成头文件.....	3
图 5-1 poll 函数大小修改示意图.....	6
图 5-2 openEuler 侧 IPC 平均值展示图.....	6
图 5-3 Baremetal 侧 IPC 平均值展示图.....	6
图 5-4 Libck 侧 IPC 平均值展示图.....	6
图 5-5 UniProton 侧 IPC 平均值展示图.....	7
图 5-6 执行命令示意图.....	7
图 5-7 执行命令示意图.....	8
图 5-8 脚本注释函数示意图.....	9
图 5-9 build_uniproton_ipc.sh 脚本注释示意图.....	9
图 5-10 UniProton 中注释示意图.....	10
图 5-11 执行命令示意图.....	10
图 7-1 sample_timer 示意图.....	13
图 10-1 注释 baremetal 中 mcs 代码示意图.....	16
图 10-2 驱动注释 mcs 示意图.....	17
图 11-1 链接脚本样例.....	18
图 11-2 XFER_DATA 数据声明样例.....	18



1 添加驱动指导

- 步骤1** 在mpu_solution/src/real_time/drivers目录下新增文件夹，文件夹名为驱动名称（以下以xxx_driver为例）。
- 步骤2** 在xxx_driver文件夹下新建 *.c、*.h、Makefile 文件（以下以xxx_driver.c为例）。
- 步骤3** Makefile中文件内容为：
- ```
obj-y +=xxx_driver.o
CFLAGS += -I./
```
- 步骤4** 修改mpu\_solution/src/real\_time/drivers目录下Makefile，在最后一行新增如下内容。
- ```
obj-y += xxx_driver/
```
- 步骤5** 在mpu_solution/src/real_time/drivers/include文件夹下新增bm_xxx_driver.h头文件，该头文件内容为对外暴露的接口及结构体等。
- 步骤6** 进入mpu_solution/build/build_baremetal目录下，执行./build_baremetal.sh即可。
- 结束



2 可用外设资源配置

本章针对于混合部署系统中驱动提供隔离使用方案指南。

图2-1 参考src/real_time/drivers/include/device_resource.h，实时侧可用的外设资源宏定义为1。



图 2-1 dts 转换成头文件

```
#define DT_N_S_SOC_S_DMAC_P_DOMAIN_CHANNEL0_EXISTS 1
#define DT_N_S_SOC_S_DMAC_P_DOMAIN_CHANNEL1_EXISTS 1
#define DT_N_S_SOC_S_DMAC_P_DOMAIN_CHANNEL2_EXISTS 1
#define DT_N_S_SOC_S_DMAC_P_DOMAIN_CHANNEL3_EXISTS 1
#define DT_N_S_SOC_S_DMAC_P_DOMAIN_CHANNEL4_EXISTS 1
#define DT_N_S_SOC_S_DMAC_P_DOMAIN_CHANNEL5_EXISTS 1
#define DT_N_S_SOC_S_DMAC_P_DOMAIN_CHANNEL6_EXISTS 1
#define DT_N_S_SOC_S_DMAC_P_DOMAIN_CHANNEL7_EXISTS 1
#define DT_N_S_SOC_S_GMAC_P_DOMAIN_GMAC0_EXISTS 1
#define DT_N_S_SOC_S_GMAC_P_DOMAIN_GMAC3_EXISTS 1
#define DT_N_S_SOC_S_GPIO_P_DOMAIN_GPIO0_EXISTS 1
#define DT_N_S_SOC_S_GPIO_P_DOMAIN_GPIO1_EXISTS 1
#define DT_N_S_SOC_S_GPIO_P_DOMAIN_GPIO2_EXISTS 1
#define DT_N_S_SOC_S_GPIO_P_DOMAIN_GPIO3_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C3_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C4_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C5_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C6_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C7_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C8_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C9_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C10_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C11_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C12_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C13_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C14_EXISTS 1
#define DT_N_S_SOC_S_I2C_P_DOMAIN_I2C15_EXISTS 1
#define DT_N_S_SOC_S_LOCALBUS_P_DOMAIN_LOCALBUS0_EXISTS 1
#define DT_N_S_SOC_S_SPI_P_DOMAIN_SPI1_EXISTS 1
#define DT_N_S_SOC_S_TIMER_P_DOMAIN_TIMER3_EXISTS 1
#define DT_N_S_SOC_S_TIMER_P_DOMAIN_TIMER4_EXISTS 1
#define DT_N_S_SOC_S_TIMER_P_DOMAIN_TIMER5_EXISTS 1
#define DT_N_S_SOC_S_UART_P_DOMAIN_UART2_EXISTS 1
#define DT_N_S_SOC_S_UART_P_DOMAIN_UART3_EXISTS 1
#define DT_N_S_SOC_S_UART_P_DOMAIN_UART4_EXISTS 1
#define DT_N_S_SOC_S_WATCHDOG_P_DOMAIN_WATCHDOG2_EXISTS 1
#define DT_N_S_SOC_S_WATCHDOG_P_DOMAIN_WATCHDOG3_EXISTS 1
#define DT_N_S_SOC_S_WATCHDOG_P_DOMAIN_WATCHDOG4_EXISTS 1
#endif /* __DEVICE_RESOURCE_H__ */
```

例：如要使用GMAC2，需在头文件中定义。

```
#define DT_N_S_SOC_S_GMAC_P_DOMAIN_GMAC2_EXISTS 1
```



3 I2C 、SPI、DMAC 使用 FAQ

1. I2C、SPI、DMAC异步接口需要用户注册一个callback。
2. callback需要用户自己实现，callback原型为(以I2C举例):

```
typedef void (*bm_i2c_callback)(bm_i2c_ids, bm_xxx_transmit_async_t);
```
3. callback示例:

```
static void i2c_transmit_async_callback(bm_i2c_ids i2c_id, bm_i2c_transmit_async_t type)
{
    printf( "transmit_id = %d, transmit_status = %d\n" , bm_i2c, type);
}
```
4. 在传输完成或者传输失败时，参数bm_i2c_ids 和bm_i2c_transmit_async_t由中断函数传入，用户通过判定bm_i2c_ids来确认传输的通道，通过bm_i2c_transmit_async_t来判定传输状态。
5. 在I2C和SPI中：
传输状态值0表示写成功；
传输状态值1表示写失败；
传输状态值2表示读成功；
传输状态值3表示读失败。
在DMAC中：
传输状态值0表示写成功；
传输状态值1表示写失败。



4 NET 使用 FAQ

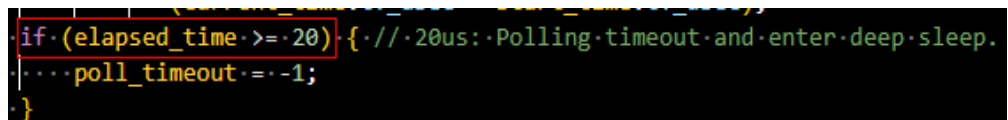
1. 用户新增网络phy芯片时，需自行适配网络phy驱动。
2. 用户使用网络驱动时，需自行保证网卡mac地址唯一性(在调用bm_net_init接口后，通过bm_net_set_mac_addr接口设置网卡mac地址)。
3. `int bm_net_set_mac_addr(bm_eth eth, const unsigned char *mac_addr);` 调用该接口设置mac地址时需要用户保证mac_addr指向的数组大于6个字节。

5 IPC 性能测试方法

特别说明：

1. open_source/mcs/library/rpmsg_endpoint.c中rpmsg_service_receive_loop函数中画红色方框地方建议修改值50，如果单次发送数据量过大，可以改为更大的数值；修改画红色方框部分如图5-1。
4字节、8字节、16字节：建议数值改为30。
32字节、64字节、128字节：建议改为60。
256字节：建议数值改为70。

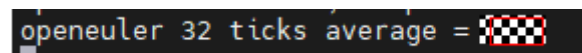
图 5-1 poll 函数大小修改示意图



```
if (elapsed_time >= 20) { // 20us: Polling timeout and enter deep sleep.  
    ... poll_timeout = -1;  
}
```

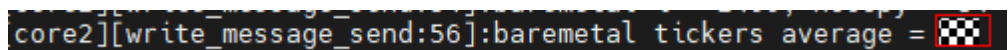
2. 计算方式：ticker = openEuler侧减去baremetal（或者其他实时侧）侧（重复三次，每次来回2499个数据）[画红色矩形框的数据]，单位为tick。
时间：ticker* 40 / 1000，单位微秒。

图 5-2 openEuler 侧 IPC 平均值展示图



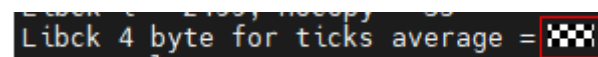
```
openeuler 32 ticks average = 100
```

图 5-3 Baremetal 侧 IPC 平均值展示图



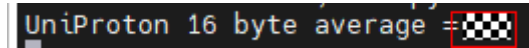
```
core2][write_message_send:56]:baremetal tickers average = 100
```

图 5-4 Libck 侧 IPC 平均值展示图



```
Libck 4 byte for ticks average = 100
```

图 5-5 UniProton 侧 IPC 平均值展示图



3. 针对Baremetal与Libck进行性能测试时，需要参考[串口打印](#)章节打开串口打印功能。

5.1 Baremetal 测试方法

前提条件：按照《openEuler+Baremetal混合部署方案编译运行指南》2.2 制作openEuler+Baremetal镜像章节内容完成制作openEuler+Baremetal镜像。

执行如下步骤：

- 步骤1** 执行如下命令编译sample程序，生成组件位于build/build_hi3093/output目录下。

```
cd mpu_solution/build
source build_prepare.sh
source build_exports
cd build_hi3093
source ~/hi3093_tool/toolchain/environment-setup-aarch64-openeuler-linux
./build_ipc_baremetal.sh
```

- 步骤2** 修改rpmsg_baremetal_ipc、mcs_core2.bin、mcs_core3.bin执行权限。

```
chmod 755 rpmsg_baremetal_ipc
chmod 755 mcs_core2.bin
chmod 755 mcs_core3.bin
```

- 步骤3** 运行sample程序。

```
./rpmsg_baremetal_ipc -i 2 -c 2 -t mcs_core2.bin -a 0x93000000 -m mcs_core3.bin -n 0x93100000
```

- 步骤4** 根据如[图5-6](#)所示打印信息，执行如下步骤。

1. 使用ssh连接192.168.0.11。
2. 执行：screen /dev/pts/2 //根据图中绿色圈里面连接console对应的pts设备。
3. 在/dev/pts/2终端中，若出现如下信息(You are not the owner of /tmp/uschreens/S-root)，则执行以下命令。
chown \${USER}:\${USER} -R /tmp/uschreens/S-root
chmod 700 /tmp/uschreens/S-root

图 5-6 执行命令示意图



----结束

说明

如果需要修改其他字节测试，按照如下方式修改sample代码：

- src/samples/non_real_time/ipc/common/sample_ipc_common.c中pty_endpoint_cb函数中chars_len的大小。
- src/samples/real_time/baremetal/ipc/sample_ipc.c中sample_mcs_cb函数中chars_len参数大小与pty_endpoint_cb中chars_len保持一致。

5.2 Libck 测试方法

前提条件：按照《openEuler+libck混合部署方案编译运行指南》2.2 统一构建镜像章节内容完成制作openEuler+libck镜像。

执行如下步骤：

步骤1 执行如下命令编译sample程序，生成组件位于build/build_hi3093/output目录下。

```
cd mpu_solution/build
source build_prepare.sh
source build_exports
cd build_hi3093
source ~/hi3093_tool/toolchain/environment-setup-aarch64-openeuler-linux
./build_ipc_libck.sh
```

步骤2 修改rpmsg_libck_ipc与mcs_core3.bin执行权限。

```
chmod 755 rpmsg_libck_ipc
chmod 755 mcs_core3.bin
```

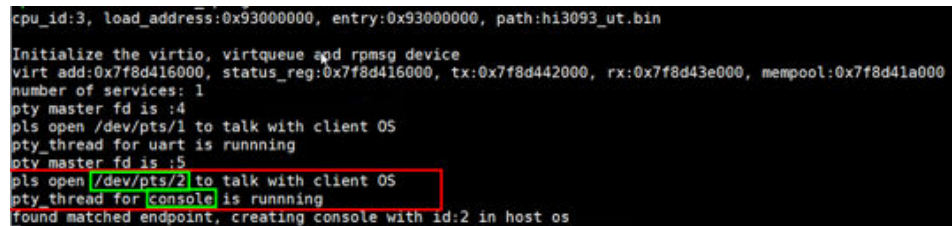
步骤3 运行sample程序。

```
./rpmsg_libck_ipc -c 3 -t mcs_core3.bin -a 0x93000000
```

步骤4 根据如图5-7所示打印信息，执行如下步骤。

1. 使用ssh连接192.168.0.11。
2. 执行: screen/dev/pts/2 //根据上图中绿色圈里面连接console对应的pts设备。
3. 在/dev/pts/2终端中，若出现如下信息(You are not the owner of /tmp/uscreeens/S-root)，则执行以下命令。
chown \${USER}:\${USER} -R /tmp/uscreeens/S-root
chmod 700 /tmp/uscreeens/S-root

图 5-7 执行命令示意图



```
cpu_id:3, load_address:0x93000000, entry:0x93000000, path:hi3093_ut.bin
Initialize the virtio, virtqueue apd rpmsg device
virt add:0x7f8d416000, status_reg:0x7f8d416000, tx:0x7f8d442000, rx:0x7f8d43e000, mempool:0x7f8d41a000
number of services: 1
pty master fd is :4
pls open /dev/pts/1 to talk with client 05
pty_thread for uart is running
pty master fd is :5
pls open /dev/pts/2 to talk with client 05
pty_thread for console is running
found matched endpoint, creating console with id:2 in host os
```

----结束

须知

如果需要修改其他字节测试，按照如下方式修改sample代码：

- mpu_solution/src/samples/non_real_time/ipc/common/sample_ipc_common.c中pty_endpoint_cb函数中chars_len的大小。
- mpu_solution/src/samples/real_time/libck/ipc/sample_ipc.c中sample_mcs_cb函数中chars_send_len参数大小与pty_endpoint_cb中chars_len保持一致。

5.3 UniProton 测试方法

前提条件：按照《openEuler+UniProton混合部署方案编译运行指南》2.2 统一构建镜像章节内容完成制作openEuler+UniProton镜像。

执行如下步骤：

步骤1 执行如下命令编译sample程序，生成组件位于build/build_hi3093/output目录下。

```
cd mpu_solution/build
source build_prepare.sh
source build_exports
cd build_hi3093
source ~/hi3093_tool/toolchain/environment-setup-aarch64-openeuler-linux
./build_ipc_uniproton.sh
```

1. 如果多次执行build_ipc_uniproton.sh时，需要注释脚本中，如图5-8所示。

./build_ipc_uniproton.sh脚本中需要注释部分：如图5-9所示/

build_ipc_uniproton.sh脚本中需要注释部分。

图 5-8 脚本注释函数示意图

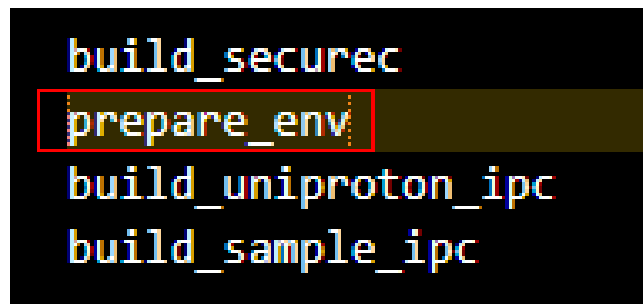
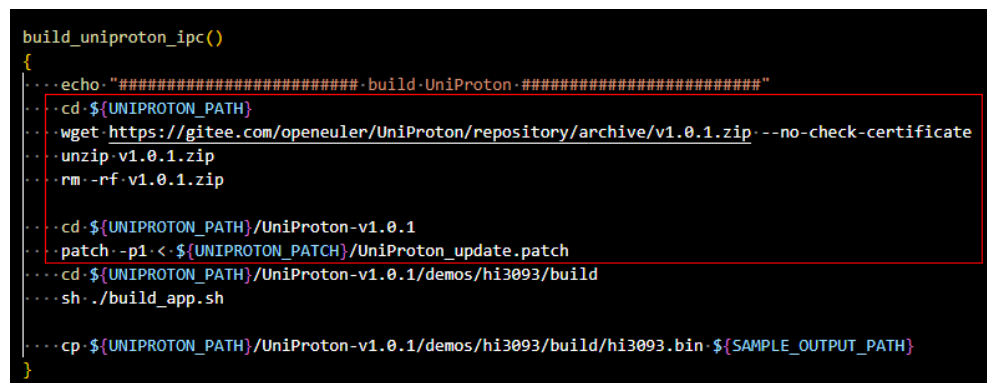


图 5-9 build_uniproton_ipc.sh 脚本注释示意图



```
build_uniproton_ipc()
{
...echo "##### build-UniProton #####"
...cd ${UNIPROTON_PATH}
...wget https://gitee.com/openeuler/UniProton/repository/archive/v1.0.1.zip --no-check-certificate
...unzip v1.0.1.zip
...rm -rf v1.0.1.zip
...cd ${UNIPROTON_PATH}/UniProton-v1.0.1
...patch -p1 < ${UNIPROTON_PATCH}/UniProton_update.patch
...cd ${UNIPROTON_PATH}/UniProton-v1.0.1/demos/hi3093/build
...sh ./build_app.sh
...cp ${UNIPROTON_PATH}/UniProton-v1.0.1/demos/hi3093/build/hi3093.bin ${SAMPLE_OUTPUT_PATH}
}
```

2. 同时还需要注释掉mpu_solution/src/real_time/UniProton/UniProton-v1.0.1/demos/hi3093/build中build_openamp.sh脚本中如图5-10画红色方框部分。

图 5-10 UniProton 中注释示意图

```
#.echo "#####.git.clone.libmetal.#####"
#.pushd ../component
#.rm -rf ./libmetal*
#.wget https://github.com/OpenAMP/libmetal/archive/refs/tags/v2022.10.0.tar.gz --no-check-certificate
#.tar -zxvf v2022.10.0.tar.gz
#.mv ./libmetal-2022.10.0 ./libmetal
#.rm v2022.10.0.tar.gz
#.cp UniProton-patch-for-libmetal.patch ./libmetal
#.cd libmetal
#.patch -p1 -d...<UniProton-patch-for-libmetal.patch
#.popd

#.echo "#####.git.clone.openamp.#####"
#.pushd ../component
#.rm -rf ./open-amp*
#.wget https://github.com/OpenAMP/open-amp/archive/refs/tags/v2022.10.1.tar.gz --no-check-certificate
#.tar -zxvf v2022.10.1.tar.gz
#.mv ./open-amp-2022.10.1 ./open-amp
#.rm -rf v2022.10.1.tar.gz
#.cp UniProton-patch-for-openamp.patch ./open-amp
#.cd open-amp
#.patch -p1 -d...<UniProton-patch-for-openamp.patch
#.cd ${BUILD_DIR}/UniProton-v1.0.1
#.patch -p1 < ${BUILD_DIR}/../patches/uniproton/uniproton_patch_for_update.patch
#.cd ${BUILD_DIR}/UniProton-v1.0.1/demos/hi3093/component/open-amp
#.popd
```

3. 如果需要修改其他字节测试，按照如下方式修改sample代码：

- mpu_solution/src/samples/non_real_time/ipc/common/sample_ipc_common.c中pty_endpoint_cb函数中chars_len的大小；
- mpu_solution/src/real_time/uniproton/UniProton-v1.0.1/demos/hi3093/apps/openamp/rpmsg_service.c中rpmsg_endpoint_cb函数中data_len参数大小与pty_endpoint_cb中chars_len保持一致。

步骤2 修改rpmsg_libck_ipc与hi3093.bin执行权限。

```
chmod 755 rpmsg_uniproton_ipc
chmod 755 hi3093.bin
```

步骤3 运行sample程序。

```
./rpmsg_uniproton_ipc -c 3 -t hi3093.bin -a 0x93000000
```

步骤4 根据如图5-11所示打印信息红色方框中的内容，执行如下步骤。

- 使用ssh连接192.168.0.11。
- 执行命令：screen /dev/pts/2 //根据上图红色圈里面连接console对应的pts设备
- 在/dev/pts/2终端中，若出现如下信息(You are not the owner of /tmp/usccreens/S-root)，则执行以下命令：

```
chown ${USER}:${USER} -R /tmp/usccreens/S-root
chmod 700 /tmp/usccreens/S-root
```

图 5-11 执行命令示意图

```
qemu-aarch64 /lib/modules # ./rpmsg_uniproton_ipc -c 3 -t hi3093.bin -a 0x93000000
cpu:3, path:hi3093.bin

Initialize the virtio, virtqueue and rpmsg device
virt add:0x7f9f25d000, status_reg:0x7f9f25d000, tx:0x7f9f289000, rx:0x7f9f285000, mempool:0x7f9f261000
number of services: 1
start client os
pty master fd is :4
pls open /dev/pts/1 to talk with client OS
pty_thread for console is running
found matched endpoint, creating console with id:2 in host os
found matched endpoint, creating shutdown with id:1 in host os
```



----结束



6 虚拟串口使用

在mpu_solution/src/real_time/libck/include/sre_common.h中需要修改如下代码：

```
#define CONFIG_VIRTUAL_SERIAL 1
```



7 下电释放资源

混合部署下电前需要用户自行去实现weak函数mcs_peripherals_shutdown关闭所用资源，如[图7-1](#)所示。

图 7-1 sample_timer 示意图

```
void mcs_peripherals_shutdown(void)
{
    #if defined(__BAREMETAL__)
        bm_disable_irq(TIMER_BASE_IRQ + TIMER_ID_3);
    #elif defined(__LIBCK__)
        SRE_HwiDisable(TIMER_BASE_IRQ + TIMER_ID_3);
    #endif
    return;
}
```



8 串口打印

libck、baremetal、uniproton的打印函数最大支持512字符长度的字符串语句。

libck和baremetal的串口功能默认没有打开，如需使用，需要更改src/real_time/drivers/include/bm_uart.h中的宏定义为：

```
#define BM_OPEN_UART_DEVICE 1
```

打印示例代码如下：

```
bm_log("this is a sample\n");
```



9 openEuler_tcpdump 编译

步骤1 从以下路径下载openEuler-22.03-LTS-SP3交叉编译链到~/hi3093_tool目录。

```
https://mirror.truenetwork.ru/openeuler/openEuler-22.03-LTS-SP3/embedded_img/aarch64/qemu-aarch64/openeuler-glibc-x86_64-openeuler-image-aarch64-qemu-aarch64-toolchain-22.03-LTS-SP3.sh
```

步骤2 执行如下命令。

```
sh openeuler-glibc-x86_64-openeuler-image-aarch64-qemu-aarch64-toolchain-22.03-LTS-SP3.sh -d /home/${USER}/hi3093_tool/toolchain/ -y \
```

步骤3 配置openEuler交叉编译环境，执行如下命令。

```
./home/${USER}/hi3093_tool/toolchain/environment-setup-aarch64-openeuler-linux
```

须知

路径前是“点(.) 空格()”，点和空格都不能少。

步骤4 编译tcpdump。

```
cd mpu_solution/build/build_tools  
./build_tcpdump.sh
```

mpu_solution/build/build_tools/output目录生成tcpdump文件。

须知

执行机上需先安装pkg-config工具（sudo apt install pkg-config ）。

----结束

10 混合部署实时侧驱动测试方法

10.1 混合部署 Baremetal 侧驱动测试

本小节描述混合部署中Baremetal侧的驱动测试方法，由于驱动测试过程中不涉及端点的创建，因此需要注释mpu_solution/open_source/mcs/rpmsg_pty_baremetal目录下rpmsg_main.c中load_client_os函数中部分，注释部分如[图10-1](#)所示。

图 10-1 注释 baremetal 中 mcs 代码示意图

```
...  
...ret = rpmsg_app_master(&client_os);  
...if (ret) {  
...    printf("rpmsg_app_master_failed: %d\n", ret);  
...    openamp_deinit(&client_os);  
...    return ret;  
...}
```

注释完成后，需要进入mpu_solution/build/build_baremetal中执行build_mcs.sh；rpmsg_main生成件位于mpu_solution/build/build_baremetal/output/mcs_rpmsg中。

10.2 混合部署 Libck 侧驱动测试

本小节描述混合部署中Libck侧的驱动测试方法，由于驱动测试过程中不涉及端点的创建，因此需要注释mpu_solution/open_source/mcs/rpmsg_pty_libck目录下rpmsg_main.c中main函数中部分，注释部分如[图10-2](#)所示。

图 10-2 驱动注释 mcs 示意图

```
...}  
...ret = rpmsg_app_master(&client_os);  
...if (ret) {  
...    printf("rpmsg_app_master_failed: %d\n", ret);  
...    openamp_deinit(&client_os);  
...    return ret;  
...}
```

注释完成后，需要进入mpu_solution/build/build_libck中执行build_mcs.sh；
rpmsg_main生成件位于mpu_solution/build/build_libck/output/mcs_rpsmg中。

11

实时侧驱动数据传输接口地址保护开关

由于网络安全需要，避免驱动数据传输接口被恶意利用导致系统挂死，所有驱动数据传输接口默认会校验输入数据地址，保证源地址和目的地址都在数据传输专用的数据段上xfer_segment，客户应用程序的链接脚本必须显式定义该数据段（如图11-1所示），并在声明待传输数据时加上XFER_DATA字段（如图11-2所示）。

用户也可以手动关闭此项地址校验：在src/real_time/drivers/include/bm_common.h中将BM_ADDR_PROTECT删除掉。

图 11-1 链接脚本样例

```
.bss (NOLOAD) :
{
    . = ALIGN(8);
    __bss_start__ = .;
    __bss_start = .;
    *(.bss)
    *(.bss.*)
    *(COMMON)
    . = ALIGN(8);
    __bss_end__ = .;
    __bss_end = .;
} > IMU_SRAM

.xfer_segment :
{
    __xfer_segment_start = .;
    *(.xfer_segment*)
    __xfer_segment_end = .;
    ASSERT((__xfer_segment_end - __xfer_segment_start) < 0x10000, "xfer sections too big!");
} > IMU_SRAM

.mmu.table.base :
{
    PROVIDE (g_mmu_page_begin = .);
    PROVIDE (g_mmu_page_end = g_mmu_page_begin + 0x8000);
} > MMU_MEM
```

图 11-2 XFER_DATA 数据声明样例

```
10 #define I2C_ID I2C_ID_3
11 #define TRANS_ADDR 0x53
12 #define BUFF_LEN 1000
13 #define I2C_TIME_OUT 1000
14 #define I2C_TRANSMIT_SYNC 1
15
16 XFER_DATA unsigned char g_buff_write[BUFF_LEN] = { 0 };
17 XFER_DATA unsigned char g_buff_read[BUFF_LEN] = { 0 };
18
```



12 uboot 控制台读写 eMMC 和 SFC

12.1 sfc 读写

以下使用以Hi3093单板u12为例：

1. 驱动加载
sf probe 0:0
2. 擦除
sf erase offset len 从flash的offset位置擦除len个字节，len必须是0x10000也就是64KB的整数倍
3. 读取flash内容到内存指定地址
sf read addr offset len 从flash的offset位置读取len个字节到内存addr位置
4. 从内存指定地址写内容到flash
sf write addr offset len 从内存的addr位置开始向flash的offset位置写len个字节

12.2 eMMC 读写

以下使用以usr区为例，block大小为512字节，十六进制需以0x开头输入，否则视为十进制：

1. eMMC初始化
mmc_init。
2. 分区切换
mmc_switch 0 切换至eMMC user分区。（0为分区标号,0:user,1-2:boot,3:RPMB,4-7:GPP）
3. 读取emmc内容到内存指定地址
mmc_rd addr offset len 从eMMC编号为offset的block开始读取len个block到内存addr位置。
4. 从内存指定地址写内容到eMMC
mmc_wr addr offset len 从内存addr位置开始向eMMC的offset编号block开始写len个block。



13 其他测试

不提供JTAG和USB接口眼图测试工具。